2010 IEEE/ACM International Conference on Green Computing and Communications & 2010 IEEE/ACM International Conference on Cyber, Physical and Social Computing

Multi-objective Virtual Machine Placement in Virtualized Data Center Environments

Jing Xu and José A. B. Fortes
Advanced Computing and Information Systems Laboratory
Department of Electrical and Computer Engineering, University of Florida
PO Box 116200, Gainesville, FL, 32611-6200, USA
Email: {jxu, fortes}@acis.ufl.edu

***Abstract*—Server consolidation using virtualization technology has become increasingly important for improving data center efficiency. It enables one physical server to host multiple independent virtual machines (VMs), and the transparent movement of workloads from one server to another. Fine-grained virtual machine resource allocation and reallocation are possible in order to meet the performance targets of applications running on virtual machines. On the other hand, these capabilities create demands on system management, especially for large-scale data centers. In this paper, a two-level control system is proposed to manage the mappings of workloads to VMs and VMs to physical resources. The focus is on the VM placement problem which is posed as a multi-objective optimization problem of simultaneously minimizing total resource wastage, power consumption and thermal dissipation costs. An improved genetic algorithm with fuzzy multi-objective evaluation is proposed for efficiently searching the large solution space and conveniently combining possibly conflicting objectives. A simulation-based evaluation using power-consumption and thermal-dissipation models (learned from profiling of a BladeCenter), demonstrates the good performance, scalability and robustness of our proposed approach when compared with four well-known bin-packing algorithms and two single-objective approaches. Our approach can seek and find solutions that exhibit good balance among the conflicting objectives while others cannot.**

I. INTRODUCTION

Today's data centers benefit from virtualization technology [1][45], which allows resources of a single server to be sliced into multiple isolated execution environments deployed on virtual machines (VMs). The net effect is fewer physical servers with much higher per-server utilization, increasing flexibility and availability, while reducing hardware costs and operational expenses related to power, cooling, physical space, etc. However, the flexibility enabled by virtualization introduces new management challenges since large pools of logical servers (i.e. VMs) must be provisioned and managed.

In the scenario considered in this paper, a virtualized data center provides a shared hosting infrastructure to customers who need resources to run their applications on a virtualized platform. Each application runs inside of its own virtual machine which can be provisioned and managed on-demand. The data center manager must respond to various on-demand resource requests by determining where VMs are placed and how the resources are allocated to them. This is a time-consuming complex task that cannot be performed by human operators in a timely fashion in increasingly larger data centers. A naïve approach for management of VMs can hurt application performance if physical resources are not sufficient for the VMs to be placed, fail to achieve potential power savings, or create expensive-to-cool hot spots when workloads are concentrated into the same physical resource.

A great amount of work has been devoted to optimize the management of a data center. Earlier work mostly focuses on improving resource usage while maintaining application performance guarantees [12][23][35]. Currently, power consumption [2][9][22][27] and thermal dissipation [18] are significant contributors to data center operational costs. To reduce these costs, the use of virtualization to consolidate workloads and tune off unloaded servers has been proposed to achieve greater energy savings [3][7][18][21][24]. Work in [25][34] proposed a temperature-aware workload placement approach to minimize peak temperature. Most of the extant work focuses on only one specific aspect of management, such as minimizing power consumption, balancing thermal distribution, or maximizing resource usage. However, these may be conflicting objectives when considered all together. For example, tightly packing VMs onto a small number of servers and turning off other servers is an effective way to reduce energy costs. However, concentrating workloads on a subset of the system resources can cause heat imbalances and create hot spots, which may impact cooling costs and degrade server life and performance. An effective strategy should consider tradeoffs among all these objectives.

This paper expands on a two-level control approach already proposed in [illegible] for automating virtual resource management. Local controllers at the application level determine the amount of resources needed by an application to guarantee its performance. A global controller at the data center level determines VM placement and resource allocation. This paper focuses on the problem of initial VM placement where a number of requests for VMs are to be placed to the available physical resources. When the datacenter starts its operation, after reset/idle states, special usage regimes etc., all physical resources are unloaded. Periodically, newly arrived VM requests are collected and allocated to physical resources that are not completely used by previously allocated VMs or were freed by VMs that were de-allocated because their tasks completed or their lifetime expired. In these cases, all the

978-0-7695-4331-4/10 $26.00 © 2010 IEEE
DOI 10.1109/GreenCom-CPSCom.2010.137

requests and their resource requirements are known as well as the available physical resources -- the controller allocates all the VMs at once, trying to find the optimal allocation in accordance with the objectives and constraints. We call this “initial” placement because it may have to be modified to cope with workloads that over time change their resource requirements. VM migration can be used to facilitate this dynamic VM placement. However, the high cost incurred during migration prohibits unlimited usage of this mechanism. Dynamic VM placement is out of the scope of this paper, being the subject of ongoing work of ours [39] on how to minimize the impact of migration while satisfying other objectives and constraints.

In this paper, the VM placement is formulated as a multi-objective combinatorial optimization problem aiming to simultaneously optimize possibly conflicting objectives. The objectives include making efficient usage of multidimensional resources, avoiding hot spots, and reducing energy consumption costs. Intelligent search methods are used to find near-optimal solutions with reasonable runtime. This paper proposes to use a modified genetic algorithm for efficiently searching global optimal solutions and a fuzzy-logic based evaluation approach for combining different objectives.

The remainder of the paper is organized as follows. Section II introduces background material. Section III presents the overall framework and Section IV formulates the virtual machine placement problem and describes the details of the proposed approach. Section V presents experimental results and analyzes the performance of the proposed approach. Section VI discusses related work and Section VII concludes the paper.

II. Background

The approach to the problem considered in this paper involves two related research areas, combinatorial optimization and multi-objective optimization, which are briefly reviewed below.

A. *Combinatorial Optimization and Genetic Algorithms*

Combinatorial optimization is concerned with methods for optimization over discrete choices (typically a large number). Many combinatorial problems are classified as NP-hard, such as the well-known task scheduling, packing, and placement problems. Heuristic techniques for solving optimization problems in an approximate way have been used widely. Among them, genetic algorithms (GAs) [13] have shown success in tackling various classes of combinatorial problems.

A genetic algorithm operates on a set of encoded strings called chromosomes, each representing a solution in the search space and being assigned a fitness value that reflects the solution’s goodness with respect to the optimization objectives. A population of candidate solutions evolves to better solutions iteratively. In each iteration, referred to as a generation, a new set of strings is created by genetic operations (crossover and mutation) on the current solution pool to form the new generation. In crossover operations, two individuals (i.e. solutions) are selected as parents and portions of their strings are used to produce new strings representing new solutions. Mutation is another common genetic operator that is applied to a single chromosome in which some of the strings are randomly selected and changed.

B. *Multi-objective Optimization and Fuzzy Logic*

Multi-objective optimization permits several objectives to be optimized simultaneously. However, it is unlikely that a solution is optimal for every individual objective. The concept of the Pareto optimum [10] was introduced to represent the set of non-dominated solutions for which none of the multiple objectives can be improved without sacrificing any others. A solution is said to be Pareto optimal if all others perform worse than it with respect to at least one of the objectives.

Several methods have been developed to derive Pareto solutions, which broadly fall into the following two categories (1) methods that attempt to minimize a single objective function by combining the objectives in some prescribed functional form; (2) methods that attempt to optimize each criterion in turn, subject to constraints derived from the optimization of previously optimized criteria. These approaches require either appropriate weights or predefined preference among different objectives, which are hard to find in most cases. Fuzzy logic [42] provides a convenient way of combining conflicting objectives and expert knowledge.

Fuzzy logic emerged in the development of fuzzy-set theory by Lotfi Zadeh [42]. A fuzzy set A is characterized by assigning to each element x the degree of membership in A. One of the most important applications of fuzzy logic is the design of fuzzy rule-based systems. These systems use fuzzy rules i.e. “IF-THEN” rules whose antecedents and consequents use fuzzy-logic statements to represent the knowledge or control strategies of the system. The process of applying fuzzy rules is called fuzzy inference. For multiple objectives, fuzzy logic allows the mapping of values of different objectives into linguistic values characterizing levels of satisfaction. The linguistic values are mapped into the interval [0~1] by the membership function associated with each objective.

III. System Architecture

As stated in the introduction, in the scenario of interest, each user requests the use of one or more applications with an expected quality of service that requires a certain amount of resources. The data center responds to the request by deploying a VM dedicated to the applications and allocating required resources to it. A two-level control system is proposed to optimize the data center resource management.

A. *Two-level Control Management*

Two types of resource mapping are involved in this virtualized data center resource management -- the mapping from application workloads to resource requirements and the mapping from virtual resources to physical resources. Based on our previous work [37], a two-level control architecture (see Fig. 1) naturally supports these two mappings through local controllers at the virtual-container level and a global controller at the resource-pool level. A local controller implemented in every virtual machine is responsible for determining the amount of resources needed by an application

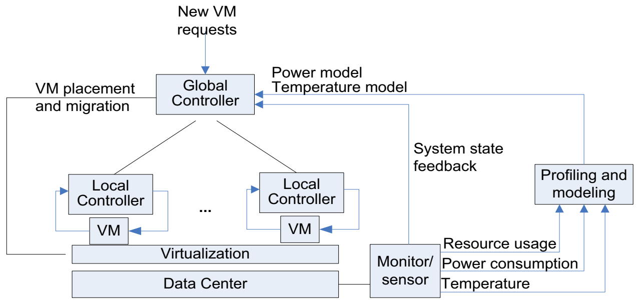

Figure 1. Two-level control architecture and information flow for autonomic resource management in a virtualized data center.

and asking for more or less resources to guarantee application performance at minimum cost. A global controller determines VM placement and resource allocation. Our previous work focused on the design of local controllers that use fuzzy logic-based modeling approaches to adaptively model the relationship between workloads and their resource demands. The evaluation through a prototype implementation demonstrates that the proposed approaches can accurately estimate resource demands for dynamically changing workloads (see details in [37]). This paper concentrates on the design of the global controller at the resource level.

The resource requests from users are expressed as VMs with specific resource needs and each of the users' applications is packaged to run on its own VM. The size of a VM is represented as a *d*-dimensional vector in which each dimension corresponds to one type of the requested resources (e.g. CPU, memory and storage). Resources on physical servers are allocated as "slices" along multiple dimensions according to the resource demands of VM requests (see an example in Fig. 2). Each VM is assigned to a slice of a server and the resources consumed by the VM are bounded by the size of this slice[1]. The monitoring system of the data center measures system information including resource usage, power consumption and temperature of each server and collects them into a centralized profiling repository. The profiling and modeling components utilize the system measurements to create models of power and temperature, which are used by the global controller to optimize its placement decisions.

B. *Placement Decision*

In essence, the global controller faces a decision-making problem. As discussed in the introduction section, the placement decision considers the following factors.

Resource Wastage: The residual resources available on each host may vary largely with different VM placement solutions. In anticipation of future requests, the resources left on each server should be balanced along different dimensions. Otherwise, unbalanced residual resources may prevent any further VM placement, thus wasting computing resources. As

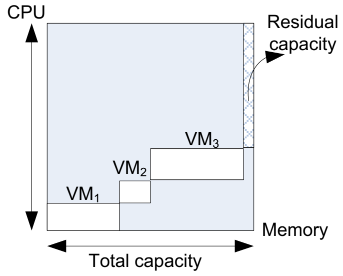

Figure 2. An example of resources allocated to three VMs placed into a single physical machine.

Fig. 2 illustrates, the outside shaded rectangle represents the total CPU and memory capacity of a physical server. The host's resource capacity is reduced along each dimension by placing three VMs and allocating resources to them. The three small rectangles indicate the amount of resources allocated to each VM. The crosshatched area in the figure denotes the residual resources available for future allocation. In the example of Fig. 2, the host has a lot of unused CPU capacity but little memory available, causing the host to not be able to accept any new VM because of memory scarcity.

To balance the resource usage along different dimensions, the following notation is used to calculate the potential cost of wasted resources. R_i represents the normalized residual resource (i.e., the ratio of residual resource to total resource) along dimension *i*. Using subscript *k* to identify the dimension that has the smallest normalized residual capacity, the wasted residual resource on a server is calculated as the sum of differences between the smallest normalized residual resource and the others, $W = \sum_{i \neq k} (R_i - R_k)$. Therefore, the bigger the differences of residual resources are among different dimensions, the more resources are wasted.

Operational Power: Power consumption tends to vary significantly with the actual computing activity. Extensive research work has been done to estimate power consumption using performance counters or system activity measurements. Based on the results from profiling power consumption of an IBM BladeCenter (see detailed experimental data in Section V), a commonly used linear power model [7][22] is used in our work to estimate the power consumption. In order to save energy, servers are turned off when they are unloaded (alternatively, low power states [16] other than "power-off" could be considered within the framework of our approach.). The total operational power (C) consumed by the servers is calculated as $C = \sum_j P_j$, $P_j = \begin{cases} p_1 + p_2 U_j^{CPU} & \text{if } U_j^{CPU} > 0 \\ 0 & \text{otherwise} \end{cases}$, where P_j and $U^{CPU}{}_j$ denote the power consumption and CPU utilization (also called CPU load) of the *j*th server.

Thermal dissipation: Thermal performance is one of the critical metrics in data center management. Sharp spikes in server utilization may result in disruptive downtimes due to generated hotspots. According to the well-known duality between heat transfer and RC circuit electrical phenomena

[1] Some existing VM resource management tools (e.g. VMware vCenter) allow setting the maximum amount of CPU or memory a VM can use.

[18], a thermal RC circuit can be used to model the steady state temperature of a server as $T = PR + T_{amb}$, where P denotes the power consumption, R denotes the thermal resistance, and T_{amb} is the ambient temperature. Using the linear relationship between power and CPU utilization, the temperature is related to the CPU load of the host according to the equation $T = (p_1 + p_2U_{CPU})R + T_{amb}$. This linear relationship between CPU temperature and CPU activity is confirmed by our profiling study of CPU temperature conducted on a BladeCenter (see data in Section V). Thermal management in data centers aims at mitigating individual hotspots and keeping temperature within a safe operating range.

Each of the above-discussed factors represents an optimization objective during VM placement. Based on the observed system states, the global controller utilizes power and thermal models to estimate the future system state and select the best placement based on the optimization criteria. Many research efforts have focused on power and thermal management leading to proposals of different power and thermal models. One advantage of our proposed global controller is that it can incorporate any type of models, depending on the investigated systems. The models used in our work were inferred from measurements on the IBM BladeCenter system mentioned in Section V.

IV. Problem Formulation

The proposed multi-objective VM placement optimization problem is formulated as follows (Table I lists the symbols used in the rest of the paper):

Goal: $\min \sum_{j=1}^{M} W_j$ and $\min \sum_{j=1}^{M} P_j$ and $\min \max(T_j)$

Constraints:

$$\sum_{i=1}^{N} r_i^{CPU} \cdot a_{ij} < c_j^{CPU},\ \sum_{i=1}^{N} r_i^{mem} \cdot a_{ij} < c_j^{mem} \quad j=[1,...,M]$$

$$\sum_{j=1}^{M} a_{ij} = 1 \qquad i=[1,...,N]$$

The first two objectives are to minimize the total resource wastage and power consumption by all the servers. The third objective function is to minimize the peak temperature among the servers. The first constraint constrains the allocated resources from each server to not exceed its capacity. The second constraint ensures that each virtual machine is allocated to one and only one of the servers. Consider N virtual machines to be placed on M available servers[2] -- there are a total of N^M possible placement solutions, making a full enumeration impractical to find the best solutions. The following shows how to apply an improved grouping genetic algorithm to efficiently search the solution space.

TABLE I. Symbols used in VM placement problem formulation

M	Number of physical servers
N	Number of VM requests
$[c_j^{CPU}\ c_j^{mem}]\ (j=[1...M])$	Capacity vector of the jth server
$[r_i^{CPU}\ r_i^{mem}]\ (i=[1...N])$	Resource requirements of the ith virtual machine
$a_{ij} \in [0,1]$	Allocation matrix in which a_{ij}=1 if vm_i is allocated to the jth server
W_j	Resource wastage of jth server
P_j	Power consumed by the jth server
T_j	Temperature of the jth server
U_{CPU}	CPU utilization
U_{mem}	Memory utilization

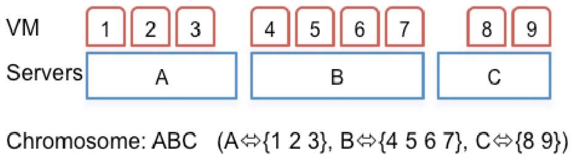

Figure 3. An example of VM placement and its corresponding chromosome

A. *Grouping Genetic Algorithm*

The grouping problem is to group a set of items into a collection of mutually disjoint subsets. Falkenauer [8] pointed out that a classic genetic algorithm (GA) performs poorly on grouping problems such as bin-packing and introduced the grouping GA (GGA), which is a GA heavily modified to suit the structure of grouping problems. A special encoding scheme is used in GGA in order to make the relevant structure of grouping correspond to genes in chromosomes. In addition, special genetic operators for crossover and mutation are used to suit the structure of chromosomes. To further improve the performance, a new operator, called ranking-crossover, is proposed and used in this work, as explained in the following.

Encoding: In grouping problems, the objective function is defined over groups rather than isolated objects. Therefore, the encoding in GGA is group oriented. Fig. 3 illustrates an example of VM placement. Nine VMs are partitioned into three VM groups (one per physical server) and the corresponding chromosome features three genes, each of them encoding a group of VMs allocated to one of three servers.

Crossover: The aim of the GGA crossover is to produce offspring out of two parents in such a way that the children inherit as much as possible of useful information from both parents. The GGA crossover randomly selects a portion of the first parent (i.e., some of the VM groups) and injects it into the second one. Some VMs could appear twice in the solution, so the groups containing them in the second parent are eliminated. Some of the VMs could be missing as a result of this elimination step -- GGA uses a local heuristic, such as first-fit, to reinsert the missing VMs.

However, this crossover operator does not perform very efficiently in our case because the inheritance is performed completely blindly with random selection and insertion, and it

[2] Current virtualization techniques already allow the creation of VMs that run across multiple servers; for simplicity this case is not considered but the proposed approach can be used by either redefining individual physical resources as collections of servers or considering multi-server VMs as multiple one-server VMs that must be co-scheduled (an additional constraint in the problem formulation).

is unlikely to obtain good results from a relatively small number of trials. A ranking-crossover is proposed to enable new generated solutions to inherit the good features from their parents more efficiently. The first step is to evaluate each individual VM group based on three types of efficiencies (discussed below) which correspond to the three aforementioned optimization objectives.

Resource usage efficiency ($E_{resource}$): It reflects how well the resources of different types are utilized. The goal is to fully utilize the resources in all dimensions. In the case of resources with CPU and memory, the efficiency is defined as the product of CPU usage and memory usage, i.e., $E_{resource} = U_{CPU}\, U_{mem}$.

Power consumption efficiency (E_{power}): It reflects how much useful work is produced under certain power consumption. $E_{power} = \frac{workload}{power} = \frac{U_{CPU}}{p_1 + p_2 \cdot U_{CPU}} \cdot (p_1 + p_2)$ (the workload is represented by the CPU utilization and the factor p_1+p_2 is used to make the efficiency value fall into [0~1] range). The power consumption efficiency increases monotonically with CPU usage, and reaches the highest point when CPU usage is 100%.

Thermal efficiency ($E_{thermal}$): A logistic function is used to calculate the thermal efficiency as $E_{thermal} = 1/(1+e^{(T\text{-}Ts)})$. The efficiency value decreases rapidly when the CPU temperature goes beyond the safe range (T_s is set to °C70 in the experiments discussed in Section V).

The values of all these three efficiencies are in the [0~1] range. The VM group evaluation function uses the average value of the three efficiencies to evaluate the groups in each solution. The next step is to compose a new solution by selecting the groups from the parents in a deceasing order of their evaluation values. When a group is selected, its VMs that appear in a previously chosen group are eliminated. In this way, the new generated solutions inherit the "good" structured groups and are able to evolve to better solutions quickly.

Mutation: GGA's mutation is also group oriented. A few VM groups are randomly selected and eliminated. The VMs in those groups are inserted back in a random order using a first-fit heuristic algorithm. However, the evaluation (see details in Section V) showed that this operation is not very useful for our case because of the blind deletion and insertion.

B. Fuzzy Multi-objective Optimization

The proposed VM placement attempts to minimize several (possibly) conflicting objectives. In order to use GGA to solve multi-objective problems, the fitness function used for selecting new generations of solutions should reflect all objectives. The solutions obtained through GGA are evaluated using the following proposed fuzzy-logic system.

Consider our VM placement problem where the goal is to minimize total resource wastage, power consumption and maximum temperature. Three linguistic variables - *resource wastage* (w), *power* (p) and *temperature* (t) - are introduced and one linguistic value and a corresponding fuzzy set are defined for each variable, namely, fuzzy sets *small resource wastage, low power consumption* and *low temperature*. Membership functions of these fuzzy sets are decreasing functions of the variable values, since the smaller the value, the higher is the degree of satisfaction. The search algorithm seeks to find solutions that are nearest to each individual goal. Hence, the evaluation of a solution can be expressed by the following fuzzy rule:

IF solution x has small resource wastage (sw), AND low power consumption (lp) AND low temperature (lt),
THEN x is a good solution.

The most desirable solution is the one with the highest membership in the fuzzy sets {sw, lp, lt}. Using the ordered weighted-averaging fuzzy operator proposed by Yager [41], the above fuzzy rule evaluates to the following, $\mu(x) = \beta\min(\mu_w(x)\ \mu_p(x)\ \mu_t(x)) + (1\text{-}\beta)\text{avg}(\mu_w(x)\ \mu_p(x)\ \mu_t(x))$ (β is set to 0.5 in the evaluation experiments described in Section V), in which $\mu_w(x)$, $\mu_p(x)$, and $\mu_t(x)$ represent the membership degree of solution x in the fuzzy sets defined by sw, lp, and lt, respectively. $\mu(x)$ is the membership value for solution x in the fuzzy set of *good solutions*. The solution with the highest $\mu(x)$ is the one that best meets all the goals and is reported as the best solution.

The membership functions for the fuzzy set {sw, lp, lt} are linear decreasing functions. The following calculation is proposed to determine the lower and upper bounds of the membership functions. The CPU and memory requirements of all VM requests are represented by R_{CPU} and R_{mem}, and the CPU and memory capacity of each physical server is c_{CPU} and c_{mem}. The ideal minimum number of servers needed to host all the VMs is $m_{min} = \max(R_{CPU}/c_{CPU}, R_{mem}/c_{mem})$ (by assuming that VMs can be partitioned and each server is fully utilized by the VMs running on it), therefore the lower bound of power consumption is $P_{lower} = m_{min}(p_1 + p_2)$. The maximum number of servers for hosting VMs is $m_{max} = \min(M, N)$, so the upper bound of power consumption is $P_{upper} = m_{max}p_1 + R_{CPU}p_2$. To determine the lower and upper bounds of *resource wastage*, we use r_i^{CPU} and r_i^{mem} to represent the percentage of CPU and memory requirements of the ith VM. The lower bound of total resource wastage is $W_{lower} = |\sum r_i^{CPU}\text{-}\sum r_i^{mem}|$ and the upper bound is $W_{upper} = \sum|r_i^{CPU}\text{-}r_i^{mem}|$. Based on the thermal model discussed in Section III, the lower and upper bounds of CPU *temperature* are $T_{lower} = p_1R+T_{amb}$ and $T_{upper} = (p_1 +p_2)R+T_{amb}$.

C. GGA with Fuzzy Multi-objective Evaluation

Figure 4 shows the major procedures of the proposed GGA algorithm with fuzzy multi-objective evaluation. The algorithm consists of two parts, randomly generating a number of solutions to form an initial population and reproducing new generations of solutions from an existing solution pool. For our problem, the initial population is produced as follows.

1. S permutations of VM request orderings are randomly generated.
2. For each VM request sequence, the first-fit algorithm is used to allocate the VMs to physical servers. In this way, S different placement solutions are generated.

During each generation of GGA a set of offspring are produced by the ranking-crossover operator discussed above. This operator ensures that the offspring inherit their parents'

```
I. Initial_GGA
   S=Population size
   $N_g$=Number of generations.
   $R_o$=Crossover rate
   $R_m$=Mutation rate
   P ← Initialization(S)              /* Generate initial population */

II. Iteration_GGA(S, $N_g$, $R_o$, $R_m$, P)
For i = 1 to $N_g$
   For j = 1 to $S*R_o$
      (x,y) = Select(P)               /* Select parents x and y randomly*/
      offspring[j] = RankingCrossover(x, y)
   Endfor

   For j = 1 to $S*R_m$
      y = Select (P)                  /* Randomly select a parent */
      offspring[j + $S*R_o$] = Mutation(y)
   EndFor
   P = EvaluateSelect(P, offspring)   /* Evaluate and select the
                                         best S solutions */
EndFor
```

Figure 4. Improved grouping genetic algorithm with fuzzy multi-objective evaluation.

important properties. The crossover and mutation rate are the percentage of new solutions generated from existing solution pool for each generation using crossover and mutation respectively. The generation selection is based the evaluation of each solution using the proposed fuzzy-logic based multi-objective evaluation. All three objectives are transformed to their corresponding fuzzy sets represented by their membership functions. By evaluating the fuzzy rule, the membership value of each placement solution is regarded as its fitness value. A number *S* of the best placement solutions are chosen from the solution pool comprising both the parents and their offspring for new generation. Therefore, the average fitness of the population and the fitness of the best individual solution increase in each generation.

V. Evaluation

The first part of this section discusses the profiling data obtained from an IBM BladeCenter, for modeling power and CPU temperature. In the rest of the section, the proposed multi-objective GGA placement algorithm is evaluated using a set of simulation experiments and compared with traditional bin-packing offline algorithms and also single-objective GGA approaches to show its performance, scalability and robustness over a wide range of environments. The simulation uses the modeling parameters obtained from profiling, so that the results can accurately capture the real system behavior.

A. Profiling and Modeling

In order to obtain the power and thermal models required by the global controller, we used IBM's advanced management module [44] to measure power and CPU temperature of Blade servers. The IBM BladeCenter has 14 HS21 blades each with two Xeon (Dual-Core) 2.33GHz processors with 4MB L2 cache, 8GB RAM, and 73GB SAS disk.

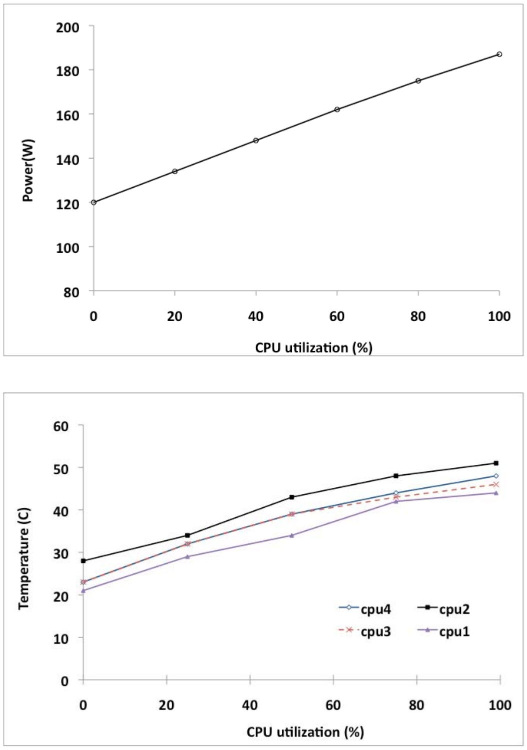

Figure 5. Power consumption (top chart) and CPU temperature (bottom chart) with varying CPU utilization.

Fig. 5(a) shows the power consumed by the blade server and Fig. 5(b) plots the CPU temperature of four cores [3] with respect to CPU utilizations. It is clear that the power consumption is linearly related to the CPU utilization and the CPU temperature is also an approximately linear function of the CPU utilization without considering the effect of heat generated by other servers nearby.

B. Evaluation of Multi-objective Placement

We use three sets of experiments to evaluate the proposed multi-objective VM placement with respect to performance, scalability and robustness. The problem size is varied by changing the number of VM requests and physical hosts. The CPU resources (measured in GHz) requested by these VMs are uniformly distributed over the set {0.25 0.5 1 1.5 2 2.5 3 4} and memory requests (measured in GB) are uniformly distributed over the set {0.25 0.5 1 1.5 2 2.5 3 4} to simulate different sizes of VM requests. Table II lists the parameters for the three sets of experiments. For each setting, we generated random inputs and ran the experiments 20 times and computed the average results (except for Fig. 7) discussed below.

1) Performance: In the first set of experiments, we compared the proposed multi-objective GGA approach with six competing algorithms including four offline bin-packing heuristics and two single-objective GGA (SGGA) approaches.

[3] The temperature measurements used for modeling use the average value of the CPU temperatures of the four cores on a server.

FFD-CPU and FFD-MEM: First-fit-decreasing (FFD) places items in a decreasing order of size, and at each step, the next item is placed to the first available bin. FFD-CPU represents the FFD solution sorted by virtual-machine CPU requirements and FFD-MEM is the FFD solution sorted by memory requirements.

BFD-CPU and BFD-MEM: Best-fit-decreasing (BFD) places a virtual machine in the fullest server that still has enough capacity. BFD-CPU and BFD-MEM represent the BFD solutions sorted by CPU requirements and memory requirements, respectively.

SGGA-P and SGGA-T: Both algorithms use GGA to search the solution space and the fitness value is evaluated with respect to power for SSGA-P, or to temperature for SSGA-T.

MGGA: Different from SGGA, the fitness value for multi-objective GGA (MGGA) is evaluated considering all the three objectives using fuzzy multi-objective optimization.

Fig. 6 compares the total resource wastage, power consumption, and maximum temperature as well as the fitness value for each of the algorithms under consideration. The key observations concerning this figure are as follows:

FFD, BFD and SGGA-P yield the highest temperature because they all tend to consolidate VMs into a smaller number of servers, resulting in higher resource utilization and higher CPU temperature. Among them, SGGA-P produces the lowest power consumption because the improved GGA is able to search the solution space more efficiently and globally so that it can find the solutions with a smaller number of used servers compared with FFD and BFD. The resource wastage of SGGA-P is also low because the placement tries to fully utilize the resources in all dimensions. On the contrary, SGGA-T tends to evenly distribute VM requests to all of the available servers, therefore the resource utilization of each server is low as well as the CPU temperature. At the same time, SSGA-T generates the highest power consumption because no servers can be turned off to save energy. MGGA produces relatively low values for power consumption, peak temperature, and resource wastage because it takes all objectives into consideration and strives to find solutions that optimize every objective and achieve good balance among conflicting goals. The best performance of MGGA compared to other competing algorithms is also confirmed by the highest fitness value shown in Fig. 6.

Fig. 7 illustrates the solution points obtained by seven placement algorithms for twenty different 128-machine 250-VM inputs using a two-dimensional graph, in which *x*-axis represents the power consumption and *y*-axis is for the CPU temperature (The dimension for resource wastage is omitted to make the figure clear.). Each point in the figure represents the solution obtained by one of the algorithms for every input. The points obtained by SGGA reside at the two ends of the figure. They either have the highest peak temperature and lowest power consumption (by SGGA-P), or lowest peak temperature and highest power consumption (by SGGA-T). The points of MGGA are located in the middle and achieve better balance between the two conflicting goals. The points obtained by FFD and BFD have the same peak temperature as SGGA-P but higher power consumption than SGGA-P, showing that they are not Pareto optimal because the solutions of SGGA-P are dominant to theirs in every objective.

TABLE II. PARAMETER SETUP FOR THREE SETS OF EXPERIMENTS

Experiment set (evaluation aspect)	Problem size *M*: No. of machines *N*: No. of VM requests	GGA parameters *S*: Initial solution size *G*: No. of generations
I (performance)	*M*=128, *N*=250	*S*=12, *G*=8
II (robustness)	*M*=128, *N*=250	*S*=[2~100], *G*=[5~20]
III (scalability)	*M*=[50~1000], *N*=[100~2000]	*S*=[5~20], *G*=[5~30]

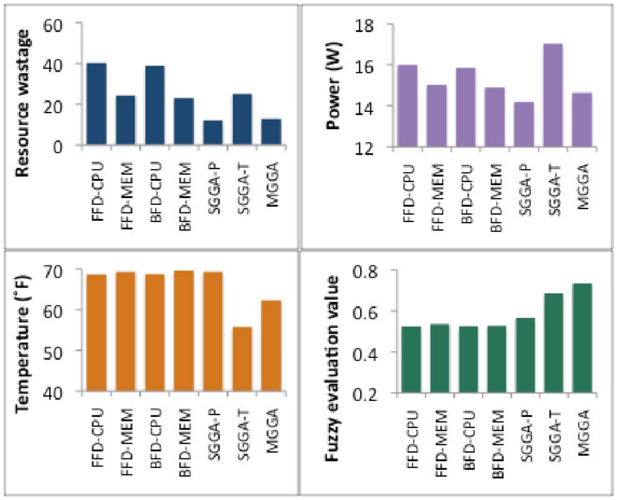

Figure 6. Performance comparisons of seven placement algorithms.

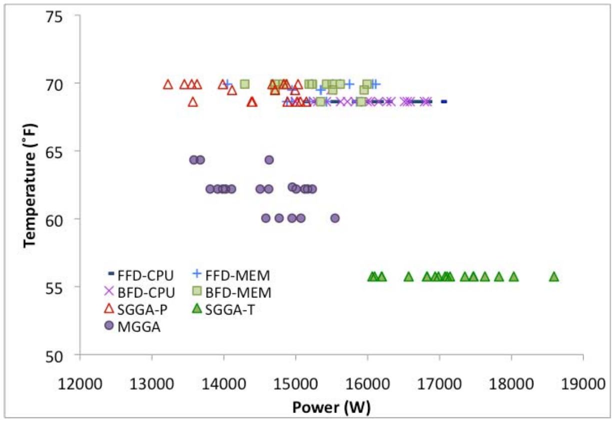

Figure 7. Solution points obtained from seven placement algorithms for twenty randomly generated 128-machine 250-VM inputs.

2) Robustness: The initial solution size (*S*) and the number of generations (*G*) are two of the fundamental parameters for the GGA algorithm. The intuition is that the performance of the algorithm improves with larger values of *S* and *G* because there are more candidate solutions to explore and more generations of solutions being produced. The previous set of experiments uses small values for both parameters, while this set of experiments explores the sensitivity of these results to the various parameter values. Fig. 8 shows the fitness values

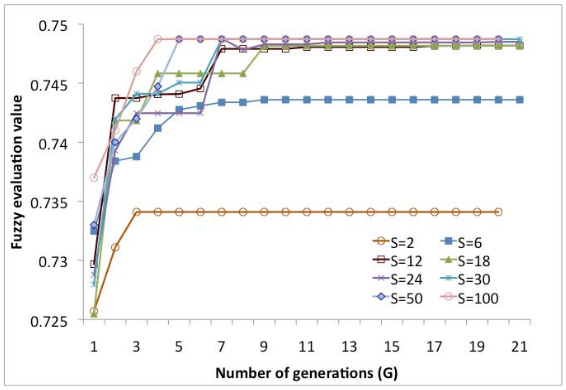

Figure 8. Fitness value of MGGA for different values of *S* and *G*.

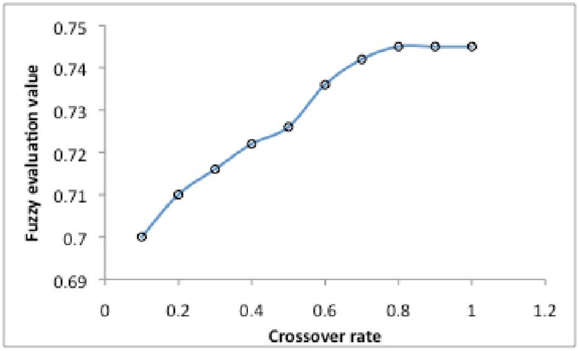

Figure 9. Fitness value of MGGA for different values of crossover rate.

for different values of *S* and *G*. When the initial solution size is very small (less than 5), the performance of GGA does not improve much even with a large number of generations because there are very few solution points available for GGA to evolve from. When the value of *S* exceeds 12, the marginal benefit of increasing initial solution size decreases rapidly, indicating that GGA has enough points to produce better solutions. Another observation is that in most cases, the performance stops improving after *G* goes beyond about 8, showing that the proposed ranking-crossover GGA algorithm can quickly improve the solutions and reach the optimal or sub-optimal points. These observations validate the robustness of the proposed GGA approach in the sense that the performance obtained by the small “representative” set of parameter values is very close to the one using larger parameter values.

As mentioned in Section IV, crossover and mutation rate are other two important parameters of the GGA algorithm. Fig. 9 plots the fitness values of GGA with varying values of crossover rate. The performance stops improving after the crossover rate goes beyond 0.8. The experiments (the results are not shown in the paper due to space limitations) also show that mutation does not help improve the performance because the random deletion and re-insertion operations does not improve the steering of solutions. Therefore, the rate of crossover and mutation are set to 0.8 and 0 for all other experiments.

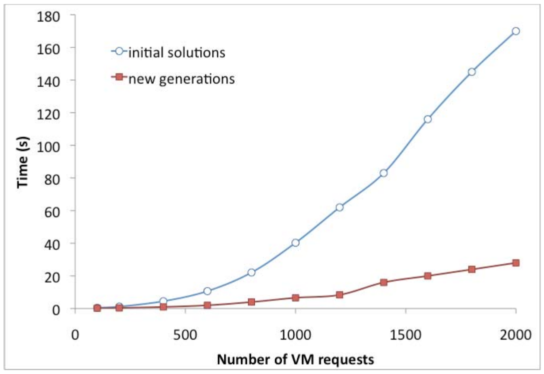

Figure 10. The execution time for producing initial population of solutions and successive generations for different problem sizes.

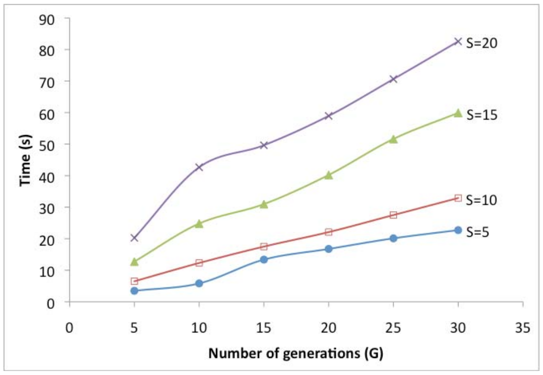

Figure 11. The execution time for generating new solutions for different values of *S* and *G*.

3) *Scalability:* The last set of experiments is used to study whether the proposed GGA algorithm is scalable for large size of data centers and VM requests. In the experiments, the number of physical machines (*M*) is varied from 50 to 1000 and the number of VM requests (*N*) from 100 to 2000. The execution time is measured on a 2GHz Pentium M machine. Fig. 10 plots the time required to generate the initial population of solutions and successive generations with increasing problem sizes. The algorithm takes less than 3 minutes to solve the difficult 1000-machine, 2000-VM placement problem. The time to generate new solutions for a 128-machine 250-VM problem with different *S* and *G* is shown in Fig. 11. It is clear that the execution time is approximately linear with respect to the values of *G* and *S*.

The complexity of the GGA algorithm consists of two parts. For the initial solution generation, the algorithm performs first-fit on a random permutation of VM requests. The complexity is O(*SNlogN*) for generating a number of *S* solutions. In the successive solution generations, the most costly function is the placement evaluation. The algorithm evaluates all the VMs placed on physical hosts in every solution, therefore the complexity is O(*NSG*) for *N* virtual machines, *S* solutions, and *G* generations. Combining these two parts, the complexity of GGA is O(*SNlogN*)+ O(*NSG*), which yields a polynomial execution time.

VI. Related Work

The problem investigated in this paper - mapping of VMs to physical servers - is related to a variety of research topics including workload placement on shared resources, dynamic resource allocation and the classical bin-packing problem.

The classical bin-packing problem is to determine how to put the items in the least number of fixed-space bins. This NP-hard problem has been extensively studied (see [6] for a recent survey). One related problem- application placement in which the goal is to maximize the total number of applications that can be hosted on a shared platform -- is theoretically studied in [36]. References [17] and [33] propose an online algorithm to dynamically place application instances into servers in response to changing resource demands. The objective is to maximize the total satisfied demands and minimize the number of placement changes. Similarly, the dynamic VM placement problem addressed in paper [15] starts with an existing mapping and the policy considered to generate new placement solutions is to balance loads among hosts. The VM placing/replacement problems are formulated as constraint satisfaction problems with the objective of minimizing the number of used servers and migration costs in [14]. A dynamic programming approach is used to solve the NP-hard problem. Much work conducted at HP labs has focused on capacity planning and workload management e.g., [4][11][12][30][43], aiming to compare and evaluate different management policies for managing a shared resource pool. Rolia et al. [29] utilized linear programming and a genetic algorithm to reduce the number of servers for placing applications in a shared resource infrastructure.

More recently, thermal and energy management have received much attention, especially for large-scale data center environments. Some work investigates the placing of applications on energy/thermal-efficient locations [19]. A temperature-aware workload placement is presented in [25][28][34]. Paper [19][31] addresses the similar problem and proposes to measure cooling efficiency for guiding the workload placement. There is also recent work on reducing power usage. Resource allocation is combined with energy management by turning off servers with no load, or low load after unloading the servers [26][37]. Some work [3][7][27][32] also considers using dynamic voltage/frequency scaling (DVFS) to further improve energy efficiency.

However, there is little prior work on workload placement and resource management that simultaneously considers efficient workload co-packing, reduced power consumption and thermal management in virtualized data center environments. The proposed approach in this paper incorporates all of these mentioned objectives and proposes a unique approach based on their evaluation using fuzzy logic.

VII. Conclusions and Future work

In this paper, the problem of VM placement is formulated as a multi-objective optimization problem aiming to simultaneously optimize possibly conflicting objectives, including making efficient usage of multidimensional resources, avoiding hotspots, and reducing energy consumption. A modified genetic algorithm is proposed and developed to effectively deal with the potential large solution space for large-scale data centers. Fuzzy multi-objective evaluation is applied in the algorithm in order to combine the conflicting goals. The profiling data obtained from measurements of an IBM BladeCenter are used for building the models of power consumption and CPU temperature of blade servers, which are then applied in the proposed placement algorithm. The simulation-based experiments studied the proposed approach with respect to its performance, scalability and robustness. The results showed the superior performance of the proposed approach compared with well-known bin-packing algorithms and single-objective approaches.

In virtualized data center environments, virtual machine migration [5] allows the transparent movement of workloads from one server to another, which can be used for adapting data center resource allocations to dynamically changing user and application demands. The dynamic VM placement approaches to cope with workloads that over time change their resource requirements are addressed in our ongoing work [39][40]. In such cases, the initial placement needs to be modified for better allocation of existing and new VMs. However, the high cost incurred by VM migration prohibits unlimited usage of this mechanism. The controller needs to minimize the impact of migration as well as satisfy other objectives and constraints, when modifying an existing placement and allocating new VMs.

Acknowledgments

This work is supported in part by National Science Foundation grants No. OCI-0721867, CNS-0821622, CNS-0855123, IIP-0758596, IIP-0932023, the BellSouth Foundation, and equipment awards from IBM. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of National Science Foundation, BellSouth Foundation, or IBM.

REFERENCES

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, “Xen and the art of virtualization,” in *Proceedings of the 19th ACM Symposium on Operating Systems Principles* (SOSP), October 2003.

[2] J. Chase, D. Anderson, P. Thakar, A. Vahdat, and R. Doyle, “Managing energy and server resources in hosting centers,” in *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, October 2001.

[3] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, “Managing Server Energy and Operational Costs in Hosting Centers,” *SIGMETRICS*, June 2005.

[4] L. Cherkasova and J. Rolia, “R-Opus: A composite framework for application performability and QoS in shared resource pools,” in *Proceedings of the International Conference on Dependable Systems and Networks (DSN'06),* 2006.

[5] C. Clark, Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., Pratt, I., and Warfield, A. “Live migration of virtual machines,” *NSDI,* 2005.

[6] E. G. Coffman Jr, M. R. Garey, and D. S. Johnson, “Approximation algorithms for bin packing: a survey,” *Approximation Algorithms for NP-Hard Problems,* PWS Publishing, Boston, 1997, 46-93.

[7] M. Elnozahy, M. Kistler, and R. Rajamony, “Energy-Efficient Server Clusters,” in *Proceedings of the Second Workshop on Power Aware Computing Systems,* February 2002.

[8] E. Falkenauer, A. Delchambre, “A Genetic Algorithm for Bin Packing and Line Balancing,” in *Proceedings of IEEE International Conference on Robotics and Automation*, 1992.

[9] X. Fan, W. Weber and L.Barroso, “Power Provisioning for a Warehouse-sized Computer,” in *Proceedings of the 34th ACM International Symposium on Computer Architecture,* June 2007.

[10] A. M. Feldman, “Welfare Economics and Social Choice Theory,” Kluwer, Boston, 1980.

[11] D. Gmach, J. Rolia, L. Cherkasova, G. Belrose, T. Turicchi and A. Kemper, "An integrated approach to resource pool management: policies, efficiency and quality metrics," *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'08),* 2008.

[12] D. Gmach, J. Rolia, L. Cherkasova, “Satisfying Service Level Objectives in a Self-Managing Resource Pool,” in *Proceedings of the Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO'2009)*, September, 2009.

[13] David Goldberg, “Genetic Algorithms in Search, Optimization and Machine Learning,” *Kluwer Academic Publishers*, Boston, MA, 1989.

[14] F. Hermenier, X. Lorca, J. Menaud, G. Muller, and J. Lawall, “Entropy: a consolidation manager for clusters,” in *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international Conference on Virtual Execution Environments* Washington, DC, March 2009.

[15] C. Hyser, B. Mckee, R. Gardner, and B.J. Watson, “Autonomic virtual machine placement in the data center,” *HP Labs Technical Report,* 2007.

[16] K. Kant and J. Alexander, “Proactive vs. Reactive Idle Power Control,” in *Proceeding of Design and Test Technology Conference*, 2008.

[17] A. Karve, T. Kimbrel, G. Pacifici, M. Spreitzer, M. Steinder, M. Sviridenko, and A. Tantawi, “Dynamic placement for clustered web applications”, *in Proceedings of the 15th International Conference on World Wide Web*, May 2006.

[18] A. Krum, F. Kreith, ed., “The CRC Handbook of Thermal Engineering,” CRC Press, 2000, pp. 2.1-2.92.

[19] D. Kusic, D. Kephart, J. Hanson, N. Kandasamy, G. Jiang, “Power and Performance Management of Virtualized Computing Environments via Lookahead Control,” in *Proceedings of 5th International Conference on Autonomic Computing* 2008.

[20] K. Le, R. Bianchini, M. Martonosi, and T. Nguyen, “Cost- and Energy-Aware Load Distribution Across Data Centers,” in *Proceedings of the Workshop on Power-Aware Computing and Systems*, October 2009.

[21] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. W. Keller, “Energy Management for Commercial Servers,” *IEEE Computer*, pp. 39-48, December, 2003.

[22] C. Lien, Y. Bai, and M. Lin, “Estimation by Software for the Power Consumption,” *IEEE Trans. on Instrumentation and Measurement*, Vol 56, No 5, October 2007

[23] X. Liu, X. Zhu, S. Singhal, and M. Arlitt, “Adaptive entitlement control of resource partitions on shared servers,” in *Proceedings of the 9th International Symposium on Integrated Network Management,* 2005.

[24] L. Mastroleon, N. Bambos, C. Kozyrakis, and D. Economou, “Autonomic power management schemes for internet servers and data centers”, In *IEEE Global Telecommunications Conference*, 2005.

[25] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, “Making Scheduling "Cool": Temperature-Aware Resource Assignment in Data Centers,” in *the 2005 Usenix Annual Technical Conference,* April 2005.

[26] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, “Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems”, In *Proceedings of the Workshop on Compilers and Operating Systems forLow Power*, September 2001.

[27] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, “No power struggles: Coordinated multi-level power management for the data center,” *ASPLOS* 2008.

[28] L. Ramos and R. Bianchini, “C-Oracle: Predictive Thermal Management for Data Centers,” in *Proceedings of the 14th International Symposium on High-Performance Computer Architecture (HPCA 14),* 2008.

[29] J. Rolia, A. Andrzejak, and M. Arlitt, “Automating Enterprise Application Placement in Resource Utilities,” in *Proceedings of the 14th IFIP/IEEE Int. Workshop on Distributed Systems: Operations and Management (DSOM)*, Heidelberg, Germany, 2003.

[30] J. Rolia, L. Cherkasova, M. Arlitt, and A. Andrzejak, “A Capacity Management Service for Resource Pools,” in *Proceedings of the 5th International Workshop on Software and Performance (WOSP),* 2005.

[31] R. Sharma, C. Bash, C. Patel, R. Friedrich, and J. Chase, “Balance of Power: Dynamic Thermal Management for Internet Data Centers,” *IEEE Internet Computing,* 9(1): 42-49, January 2005.

[32] V. Sharma, A. Thomas, T. Abdelzaher, and K. Skadron. “Power-aware QoS Management in Web Servers”, in *Proceedings of the Real-Time Systems Symposium*, December 2003.

[33] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, “A scalable application placement controller for enterprise data centers,” *in Proc. of the 16th international conference on World Wide Web*, 2007.

[34] Q. Tang, S. Gupta and G. Varsamopoulos, “Energy-Efficient, Thermal-Aware Task Scheduling for Homogeneous, High Performance Computing Data Centers: A Cyber-Physical Approach.” in *Trans. on Parallel and Distributed Systems, Spec. Issue on Power-Aware Parallel and Distributed Systems,* 2008.

[35] B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goyal, “Dynamic Provisioning of Multi-tier Internet Applications,” in *Proceedings of 2nd International Conference on Autonomic Computing,* 2005.

[36] B. Urgaonkar, A. Rosenberg, and P. Shenoy, “Application Placement on a Cluster of Servers,” *International Journal on Foundations of Computer Science (IJFCS),* Vol. 18, No. 5, October 2007.

[37] Z. Wang, M. Marwah, C. Bash, P. Ranganathan, and X. Zhu, “Delivering Energy Proportionality with Non Energy-Proportional Systems -- Optimizing the Ensemble,” in *Proceedings of the Workshop on Power Aware Computing and Systems* (HotPower '08), 2008.

[38] J. Xu, M. Zhao, J. Fortes, R. Carpenter, and M. Yousif, “On the Use of Fuzzy Modeling in Virtualized Data Center Management,” in *Proc. of 4th International Conference on Autonomic Computing,* 2007.

[39] J. Xu, J. Fortes, “A Multi-objective Cross-layer Approach to Virtual Machine Management in Virtualized Datacenters,” *submitted for publication,* 2010.

[40] J. Xu, “Autonomic Application and Resource Managed in Virtualized Distributed Computing Systems,” *Doctoral dissertation,* 2010.

[41] R. Yager, “On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decision Making”, in *IEEE Trans. on Systems, Man and Cybernetics*, 1998.

[42] L. A. Zadeh, et al. 1996 Fuzzy Sets, Fuzzy Logic, Fuzzy Systems, World Scientific Press.

[43] X. Zhu, D. Young, B. Watson, Z. Wang, J. Rolia, S. Singhal, B. McKee, C. Hyser, D.Gmach, R. Gardner, T. Christian, L. Cherkasova, “1000 Islands: An Integrated Approach to Resource Management for Virtualized Data Centers,” *J. Cluster Computing, Special Issue on Autonomic Computing*, Volume 12, Number 1, March, 2009.

[44] http://www.demos.ibm.com/servers/Demo/IBM_Demo_IBM_BladeCenter_Advanced_Management_Module-Nov06.html

[45] VMware: http://www.vmware.com/pdf/vc specs.pdf